Analog Accelerator for Simulation and Diagnostics

Alexander Feldman,[1] Ion Matei,[1] Emil Totev,[2] Johan de Kleer[1]
[1]PARC Inc., 3333 Coyote Hill Road, Palo Alto, 94304 CA, USA
e-mail: {afeldman,ion.matei, dekleer}@parc.com
[2]Philips Research, High Tech Campus 4, 5656 AE Eindhoven, The Netherlands
e-mail: emil.totev@philips.com

Abstract

We propose a new method for solving Initial Value Problems (IVPs). Our method is based on analog computing and has the potential to almost eliminate traditional switching time in digital computing. The approach can be used to simulate large systems longer, faster, and with higher accuracy.

Many algorithms for Model-Based Diagnosis use numerical integration to simulate physical systems. The numerical integration process is often either computationally expensive or imprecise. We propose a new method, based on Field-Programmable Analog Arrays (FPAAs) that has the potential to overcome many practical problems. We envision a software/hardware framework for solving systems of simultaneous Ordinary Differential Equations (ODEs) in fraction of the time of traditional numerical algorithms.

In this paper we describe the solving of an IVP with the help of an Analog Computing Unit (ACU). To do this we build a special calculus based on operational amplifiers (op-amps) with local feedback. We discuss the implementation of the ACU on an Integrated Circuit (IC). We analyze the working if the IC and simulate the dynamic Lotka-Volterra system with the de-facto standard tool for electrical simulation: SPICE.

Introduction

Analog computers have been used extensively for solving many problems. The differential analyzer (Bush 1931) is an early example. In this paper we leverage the advancement of microelectronics to propose a new implementation of a reconfigurable analog computer.

We propose an Integrated Circuit (IC) design that can dramatically improve simulation of physical systems. Our idea bridges the worlds of hybrid analog and digital electronic design, numerical methods and simulation of dynamic systems, and diagnostic and prognostic reasoning.

Traditional methods for solving systems of Ordinary Differential Equations (ODEs) involve numerical integration methods such as the classical Runge-Kutta algorithm (Butcher 1987). Many algorithms for solving ODEs are implemented in the SUNDIALS (Hindmarsh et al. 2005) library, the industrial and scientific standard for numerical integration. Depending on properties of the ODEs such as stiffness, however, methods from SUNDIALS may be slow or they may fail to converge.

Instead of evaluating the right-hand side of an integration problem as is done traditionally, our IC uses analog computing and eliminates the need of any clocks during the computation. The integration is entirely driven by the propagation of electrical signal and transistor switching time is reduced to minimum.

To map a system of simultaneous ODEs to our ACU we propose a calculus that is based on elementary computational elements, built of operational amplifiers (op-amps).

The IC design we propose is mixed analog and digital. There is a classical ARM core that is responsible for configuring the ACU. The firmware configures all transistor switches connecting the op-amps to the analog data buses. The digital core is also responsible for calibrating the analog computational elements to compensate for environmental factors such as temperature.

The approach we present in this paper faces many challenges. Real analog components behave differently from their idealized models. Electrical noise propagates alongside the computation and decreases the precision of the result. Environmental factors such as temperature variations increase the computational error. Transistor size in analog ICs is typically larger than in digital ones which makes the analog approach less competitive. Making linear resistors and digital potentiometers that are both accurate, linear, and with high resolution is an open topic of research. Power consumption of analog components is generally increased and thermal management could be a problem by itself.

In this paper, we address part of the above challenges with improved framework-level design, self-compensating mechanisms, extensive use of local feedback and novel hybrid digital/analog techniques.

Problem Definition

Our goal is to develop a method that solves a system of simultaneous Ordinary Differential Equations (ODEs):

$$\mathbf{y}^{(n)} = (F)\left(x, \mathbf{y}, \mathbf{y}', \mathbf{y}'', \ldots, \mathbf{y}^{(n-1)}\right) \tag{1}$$

where the initial values are given.

This paper uses a water clock model for illustration purposes. The water clock is a single vessel filled with liquid

(usually water) and with an orifice at the bottom for the liquid to leak out. It has been used since ancient times (Langton 1989). The water clock is modeled by the single first-order ODE:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = k\frac{a}{A}\sqrt{h} \tag{2}$$

where A is the radius at the top of the conical tank, a is the radius of the orifice, k is -4.8 (it combines several parameters such as the gravitational constant, viscosity, etc.), and h is the water level.

To turn the problem of solving an Initial Value Problem (IVP) into a diagnostic problem, we typically add one or more fault parameters, denoted as $\mathbf{f}$, to the original system, introduce a measurement function $\mathbf{H}$ (often algebraic identity), a measurement values vector $\mathbf{z}$ and solve a parameter estimation problem. There are multiple methods for solving the parameter estimation problem such as Kalman filters (Wan and van Der Merwe 2000), particle filters (Liu and Chen 1998), optimization, regression, and, even, machine-learning (Cortes and Vapnik 1995).

The water clock IVP problem is turned into a diagnostic problem by multiplying the ka/A with f_1, and assuming that under normal conditions the initial value of $f_1 = 1$ and it does not change in time:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = f_1 k\frac{a}{A}\sqrt{h} \tag{3}$$

$$\frac{\mathrm{d}f_1}{\mathrm{d}t} = 0 \tag{4}$$

Most diagnostic and parameter estimation methods solve Eq. 1 for a range of initial conditions. In general, the more instances of the above problem we solve, the more accurate the diagnostic estimation is. As a result a method to accelerate the solution of IVPs will benefit both the diagnostic accuracy and the diagnostic speed.

Framework Architecture

The proposed framework includes a software package and a hardware accelerator in the form of a PCI controller. This is to a large extent similar to the digital equivalent of an FPAA: a Field Programmable Gate Array (FPGA). In FPGA we start with a VHDL or Verilog source which is then compiled to an intermediate format. After simulation and verification, the FPGA tool-chain performs placing and routing. The hardware design is then further analyzed and flashed on to the device.

In solving IVPs with FPAAs we also have a work-flow similar to the one in FPGAs, as shown in Figure 1.

We start by building an Abstract Syntax Tree (AST) of the ODEs, and apply simplification, and constant folding.

The second work-flow step is mapping the ODEs or parts of the ODEs to computational elements. The process is similar to converting a system of ODEs to an explicit form and is related to acausal to causal conversion, and systems like MODELICA (Fritzson 2010). As our method is providing acceleration to numerical integration, this mapping is outside of the scope of our proof-of-concept analysis. We assume

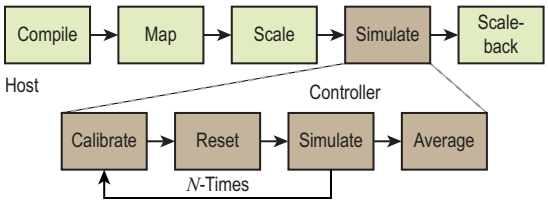

Figure 1: Framework architecture

that the ODEs will be in explicit form and solvable with integration methods. We can think of the first step of our workflow as a direct automatic mapping from ODEs, represented symbolically, to a SIMULINK model. SIMULINK is a visual modeling and simulation language that uses computational primitives, called blocks, and signals. It can use integration if needed.

The third step is rewriting the ODEs such that all state variables are within certain constraints (often between zero and one). The process can be automated and is a subject of its own (Langtangen and Pedersen 2016). The process of automated scaling involves the replacement of all variables with dimensionless variables by introducing extra parameters. We also want to change the "time constant" to make the integration faster.

At the next stage N simulations are performed with the same starting conditions. Before each simulation the device is reset, and all calibration trimmers are adjusted so that the ACU can reduce the error. This "calibration-in-the-loop" is a novel idea which is possible due to our hybrid FPAA design.

During the "reset" phase, all initial conditions are imposed. At the end of each hardware simulation, the simulation result vector is stored into memory and, at the end the hardware stage, the output of all simulation is averaged.

Figure 2 shows the technology mapping of Eq. 3 to analog elements.

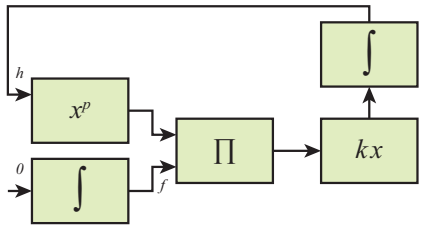

Figure 2: Technology mapping of the Clepsydra diagnostic model

The most difficult aspect of preparing an ODE for solving on the FPAA is scaling. Checking if state variables change within certain ranges can be impossible without actually solving the system. The FPAA design we propose also includes op-amp comparators to detect if values are out of range and generates events and interruptions in these cases. Whenever a state variable goes out of range during run-time, a respective parameter can be changed and the simulation restarted or continued. So, scaling the original ODE is performed both analytically, during the off-line stage, and dur-

ing simulation.

Integrated Circuit Design

Design and manufacture of mixed analog and digital integrated circuits is expensive and subject to multiple physics and technology constraints. The initial target for the proposed chip is an $180\,\mathrm{nm}$ CMOS wafer. In the future, we plan designing for the finer feature size of $40\,\mathrm{nm}$.

The design requires very quiet and stable power supply. Making bipolar operating amplifiers on IC is prohibitively expensive and difficult and we use a unipolar power supply that produces $V_{\mathrm{dd}} = 2.5\,\mathrm{V}$. The power supply is tapped in the middle and this is the common reference point of $V_{\mathrm{ref}} = 1.25\,\mathrm{V}$. The analog signals logical zero point is at V_{ref}.

The elements of the local feedback calculus are realized with two types of elements: single input and two inputs. The single input elements are gain, logarithm, exponentiation, power, integration, and differentiation. The elements with two inputs are summation and multiplication. In what follows, we discuss the working, constraints, performance, and peculiarities of each of those elements.

Op-Amp Design

Figure 3 shows a basic two-stage CMOS op-amp as described by Hussein Baher (Baher 2012).

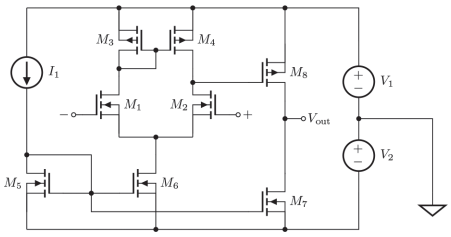

Figure 3: Classical MOSFET implementation of an op-amp

The op-amp shown in Figure 3 has two stages. MOSFETs M_1 and M_2 (both n-type) form a differential pair that drives the active load of the current mirror of M_3 and M_4 (M_3 and M_4 are both p-type). The second state of the amplifier is formed by M_7 (n-type), with M_8 (p-type) connected as an active load. Transistors M_5 and M_6 (both n-type).

Analysis of the op-amp's characteristics are beyond the scope of this paper. The basic architecture we use has to be extended to allow for precision amplification. By precision we mean both noise and non-linearities. Many noise problems can be alleviated by repeating the computation multiple times and averaging the results. Non-linearities, especially at high frequencies can be improved with a more advanced op-amp design.

Fixed Gain

Figure 4 shows one of the simplest uses of an op-amp: multiply an input voltage V_{in} with a constant k. This configuration as well as all operational amplifiers in this paper use negative feedback.

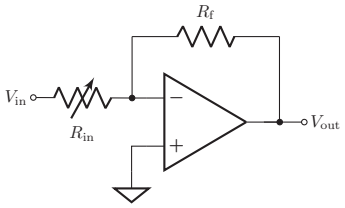

Figure 4: Inverting op-amp with negative feedback

The voltage V_{out} at the output of the inverting amplifier is:

$$V_{\mathrm{out}} = -\frac{R_{\mathrm{f}}}{R_{\mathrm{in}}} V_{\mathrm{in}} \tag{5}$$

When $R_{\mathrm{f}} = R_{\mathrm{in}}$ the negative feedback amplifier changes the sign of the input voltage V_{in}. The feedback resistor value is fixed to $R_{\mathrm{f}} = 10\,\mathrm{k\Omega}$. The value of the input resistor R_{in} can vary in the range $0\,\mathrm{k\Omega}$ to $50\,\mathrm{k\Omega}$. The plot in Figure 5 shows an LTSPICE (Brocard 2013) parameter sweep of the steady state of V_{out} for the full range of R_{in}. There are two lines: one for the universal LTSPICE model of an op-amp, and another one for the proposed op-amp design shown in Figure 3.

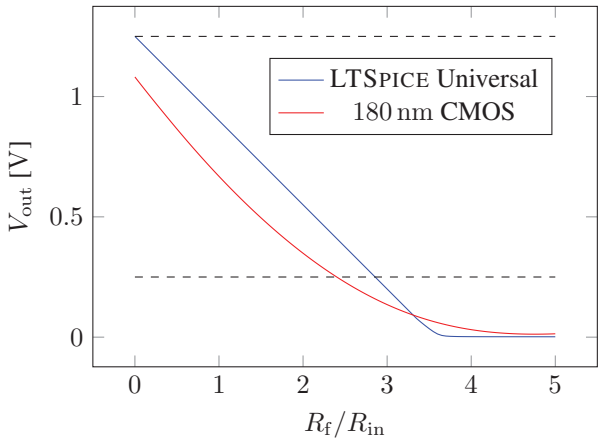

Figure 5: Parameter sweep of the inverting operational amplifier

The voltage V_{out} in Figure 5 decreases linearly until it approaches the negative power supply rail of the op-amp. As we plan to stay at least $250\,\mathrm{mV}$ away from the power-supplies, the useful range for R_{in} is from $0\,\mathrm{k\Omega}$ to $28.5\,\mathrm{k\Omega}$.

As we can see in Figure 5 the CMOS implementation of the op-map shows good linearity in the operational region $1.25\,\mathrm{V}$ to $0.25\,\mathrm{V}$ (the operational region is denoted with horizontal dashed lines). A disadvantage of the proposed CMOS design is the relative large DC offset: $\approx 200\,\mathrm{mV}$. We plan to address the DC offset issue by improving the op-amp design and by using chopper stabilization as shown by Hussein Baher (Baher 2012).

Figure 6 shows a schematic of a non-inverting op-amp. Notice that we still have negative feedback, but the negative and positive inputs of the inverting op-amp are swapped.

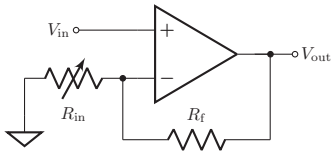

Figure 6: Non-inverting op-amp

The voltage V_{out} at the output of the non-inverting amplifier is:

$$V_{\text{out}} = \left(1 + \frac{R_{\text{f}}}{R_{\text{in}}}\right) V_{\text{in}} \tag{6}$$

The inconvenience of the non-inverting op-amp is that it only supports gain values of $A \geq 1$. Still, it can eliminate the need of an extra inverting op-amp for some ODEs. The non-inverting op-amp can be combined in a single element with the inverting op-amp with the help of a CMOS cross-over switch at the op-amp inputs.

Logarithms and Exponentiation

Figure 7 shows a logarithmic op-amp. Notice that the diode is reverse biased. The voltage and current relationship in a diode is exponential and depends on several factors such as the geometry of the p-n junction and its temperature. As the currents involved are very small (sometimes in the order of nanoamps), the standalone use of this logarithmic op-amp is not recommended. When building circuits for multiplication and raising to a power, however, logarithmic op-amps are combined with exponential op-amps (they are dual). This combination automatically provides compensation for parameter drifts and errors.

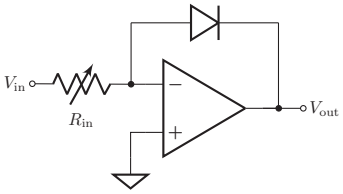

Figure 7: Logarithmic op-amp

The voltage V_{out} at the output of the logarithmic op-amp is:

$$V_{\text{out}} = -V_T \log \frac{V_{\text{in}}}{I_s R_{\text{f}}} \tag{7}$$

where V_T is the thermal voltage of the diode and I_s is the diode's saturation current. The thermal voltage is approximately $25.8563\,\text{mV}$ at $300\,\text{K}$.

The voltage V_{out} at the output of the exponential amplifier is:

$$V_{\text{out}} = -I_s R_{\text{in}} \exp\left(\frac{V_{\text{in}}}{V_T}\right) \tag{8}$$

Similarly to the inverting and non-inverting op-amps, the logarithmic and exponential configurations can be implemented with a single op-amp and a combination of CMOS analog switches that can configure the universal computational element depending on the ODE. Of course, the switching elements bring their own resistance, capacitance, and non-linearity and there is a trade-off between how configurable the FPAA element is and the precision of the computation.

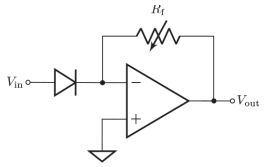

Figure 8: Exponential op-amp

Summation

The electrical circuit shown in Figure 9 is used to add two voltages: V_{x} and V_{y}. The summing op-amp is similar to the inverting one when the two input resistances match: $R_{\text{in}} = R_{\text{x}} = R_{\text{y}}$. In this case, it is easy to see that the ouput voltage is:

$$V_{\text{out}} = -\frac{R_{\text{f}}}{R_{\text{in}}} \left(V_{\text{x}} + V_{\text{y}}\right) \tag{9}$$

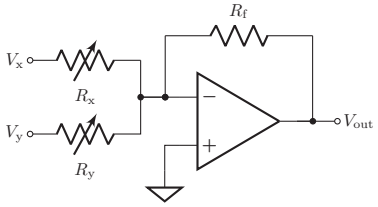

Figure 9: Adding op-amp

Multiplication and Powers

Both multiplication and raising to a power work with logarithms to avoid very large currents and voltages. Figure 10 shows how the product of the two voltages V_x and V_y is computed. First the logarithms of V_x and V_y are taken. The two results are summed by an analog adder. Finally, the output of the adder is exponentiated.

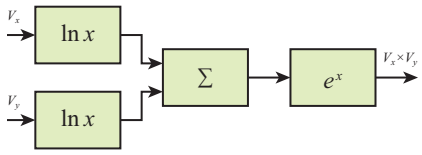

Figure 10: Multiplication with four op-amp blocks

When using the multiplication block, R_{in} of the logarithm and and R_{f} of the exponentiation have to match so they cancel each other:

$$R_{\text{f}} = \frac{R_{\text{in}}}{I_s} \tag{10}$$

Raising a number to a power is similar to multiplication in that it also uses logarithms. The op-amp implementation is shown in Figure 11.

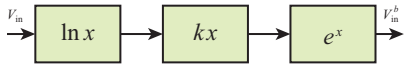

Figure 11: Raising to a power with three op-amp blocks

Similar, to the multiplication case, when using the raising to the power block, R_f of the logarithm and R_in exponentiation have to cancel each other. For this to happen, it must hold that:

$$R_\mathrm{f} = \left(\frac{R_\mathrm{in}}{I_s}\right)^p \tag{11}$$

where p is the power.

Integration and Differentiation

Figure 12 shows an integrating op-amp. Integrating op-amps are well-known and widely used in analog computing and analog-to-digital conversion.

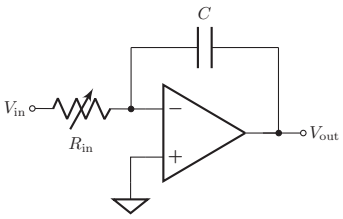

Figure 12: Integrating op-amp

The voltage V_out at the output of the integrating amplifier is:

$$V_\mathrm{out} = \frac{1}{R_\mathrm{in}C}\int_0^t V_\mathrm{in}\,\mathrm{d}t \tag{12}$$

By varying the value of R_in we can change the "time constant" of the integrating op-amp. This allows long simulations to be simulated in shorter time. Making R_in too large, though, results in high-frequency AC signals through all op-amps. Practical op-amps do not have the same gain when the frequency increases which limits the simulation speed. Making capacitors on integrated circuits is expensive as they take a lot of surface. Therefore, in our design, we will leave small capacitors (in the order of a few pico to nano-farads) on-board the IC, while larger capacitors can be implemented as capacitor arrays and used externally at the cost of a few pins.

If we swap the capacitor with the resistor of the integrating op-amp, we get a differentiating op-amp.

Simulation and Validation

We continue with a two-variable oscillating dynamic system. Consider two animal species in an ecosystem: bunnies and coyotes. The population dynamics can be modeled by the well-known Lotka-Volterra equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta xy \tag{13}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \delta xy - \gamma y \tag{14}$$

The parameters describing the interaction of the two species are $\alpha = 1$, $\beta = 5$, $\gamma = 1$, and $\delta = 3.5$. The initial values are $x = 0.1$ and $y = 0.15$. The parameters have been scaled such that all system variables are between zero and one.

Figure 13 shows the FPAA implementation of the Lotka-Volterra simulation model. The implementation uses a total of 16 op-amps. The sign inversion in the gain and summation blocks poses an inconvenience as each block requires an additional inverting op-amp.

Figure 14 shows the simulation results with the LTSPICE universal op-amp model. The two integration blocks have input resistance of $R_\mathrm{in} = 100\,\mathrm{k}\Omega$ and $1\,\mathrm{nF}$ capacitors. This results in a time scale factor of 10^3, i.e., $40\,\mathrm{s}$ of simulation time are simulated in $4\,\mathrm{ms}$.

Related Work

The closest resembling our IC and framework design is that of FPAAs. FPAAs are not new, there is a book (Pierzchala et al. 2013), and even a company specializing in designing FPAA chips.[1] What is novel in our proposal is that we specialize FPAA for simulating dynamic systems and overcome the scalability problem by inventing an op-amp negative feedback calculus.

Deese, Jimenez, and Nwankpa propose the use of FPAAs for simulating power systems (Deese, Jimenez, and Nwankpa 2009). Similar to us, the authors illustrate the applicability of their approach by simulation and prototype analysis. We have taken this research further by generalizing it to simulation and diagnostics of arbitrary dynamic systems. We have further analyzed the approach using a specific CMOS implemented op-amp

Schlottmann, Petre, and Hasler present an approach to converting SIMULINK models to FPAA configurations (Craig R Schlottmann and Hasler 2012), which are simulated using SPICE. One of the examples in this paper is a Hodgkin-Huxley-type neuron block. The results in this paper are reassuring that FPAAs can be used for the simulation of realistic and relevant dynamic systems.

Znamirowski, Palusinski, and Vrudhula have proposed to use FPAAs in simulation in control (Znamirowski, Palusinski, and Vrudhula 2004). The accent of their paper is on adaptive control which is related to diagnostics.

Conclusion

In this paper we have built a proof-of-concept for a diagnostic and simulation hardware accelerator. We have shown how an IVP can be mapped on to a grid of analog elements. We have created a special local feedback op-amp calculus

[1] http://www.anadigm.com/

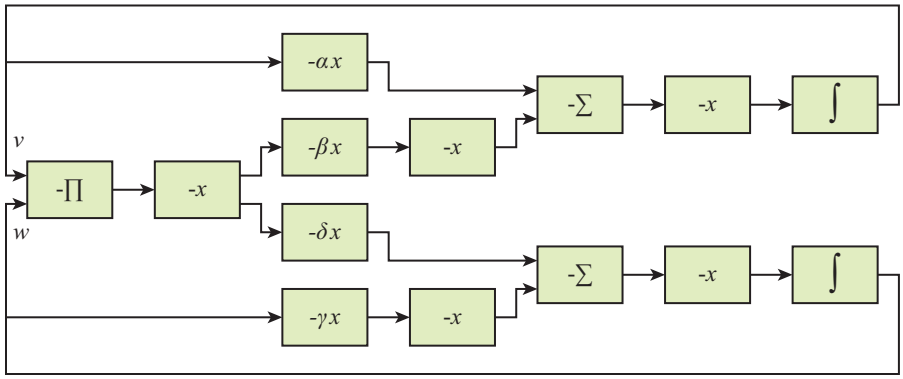

Figure 13: FPAA implementation of the Lotka-Volterra simulation model

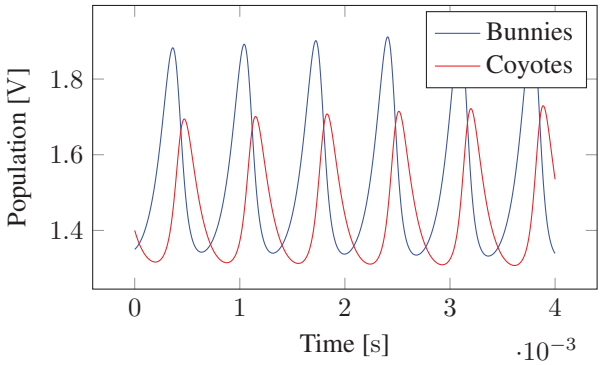

Figure 14: LTSPICE simulation of Lotka-Volterra with universal op-amps

that can implement a variety of ODEs. We have used in simulation a special CMOS op-amp for the building of the analog elements. Preliminary experiments with SPICE show that the proposed software/hardware architecture can be used for solving non-trivial oscillatory ODEs.

We next plan to improve the proposed CMOS op-amp, adding a circuit to remove the bias, to decrease the DC offset and to improve the bandwidth. We plan to analyze new types of analog elements, such as Fourier transforms. We plan to analyze the hybrid part of the IC: analog-to-digital and digital-to-analog conversion. We also plan to introduce event handling with op-amp comparators and timing and delay elements.

References

Baher, H. 2012. *Signal Processing and Integrated Circuits*. John Wiley & Sons.

Brocard, G. 2013. *The LTSpice IV Simulator: Manual, Methods and Applications*. Würth Elektronik.

Bush, V. 1931. The differential analyzer. a new machine for solving differential equations. *Journal of the Franklin Institute* 212(4):447–488.

Butcher, J. C. 1987. The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Craig R Schlottmann, C. P., and Hasler, P. E. 2012. A high-level simulink-based tool for FPAA configuration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20(1):10–18.

Deese, A.; Jimenez, J. C.; and Nwankpa, C. O. 2009. Utilization of field programmable analog arrays (fpaa) to emulate power system dynamics. In *Proceedings of The IEEE International Symposium on Circuits and Systems, 2009. ISCAS'2009*, 1713–1716. IEEE.

Fritzson, P. 2010. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. John Wiley & Sons.

Hindmarsh, A. C.; Brown, P. N.; Grant, K. E.; Lee, S. L.; Serban, R.; Shumaker, D. E.; and Woodward, C. S. 2005. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)* 31(3):363–396.

Langtangen, H. P., and Pedersen, G. K. 2016. *Scaling of Differential Equations*, volume 2. Springer.

Langton, C. G. 1989. Artificial life.

Liu, J. S., and Chen, R. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association* 93(443):1032–1044.

Pierzchala, E.; Gulak, G.; Chua, L.; and Rodríguez-Vázquez, A. 2013. *Field-Programmable Analog Arrays*. Springer Science & Business Media.

Wan, E. A., and van Der Merwe, R. 2000. The unscented Kalman filter for nonlinear estimation. In *The IEEE 2000 on Adaptive Systems for Signal Processing, Communications, and Control (AS-SPCC'2000)*, 153–158.

Znamirowski, L.; Palusinski, O. A.; and Vrudhula, S. B. 2004. Programmable analog/digital arrays in control and simulation. *Analog Integrated Circuits and Signal Processing* 39(1):55–73.